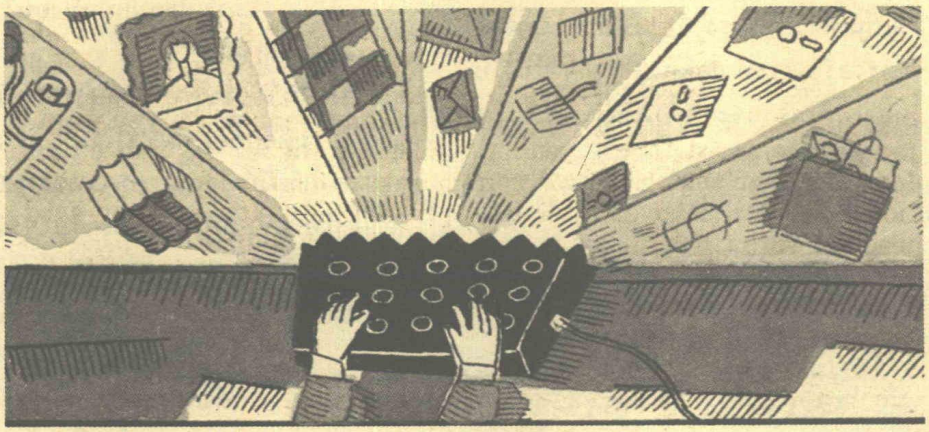
ภาษาเครื่องสำหรับภาษาสูง ตอน 5

เขียนโปรแกรมภาษาเครื่องเป็นโครงสร้าง



บุญเลิศ เอี่ยมทัศนา*

ดังได้เคยกล่าวแล้วว่าการเขียนภาษาเครื่องในปัจจุบันนิยมเขียนเลียนแบบภาษาสูงที่เป็นโครงสร้าง เช่น ปาสคาล และซี ในฉบับนี้จึงจะกล่าวถึงการเขียนโปรแกรมเป็นโครงสร้าง ซึ่งในภาษาสูงข้อกำหนดของภาษานั้น ๆ จะบังคับให้เขียนโปรแกรมเป็นโครงสร้างไปในตัว

แต่ในภาษาเครื่องเราจะต้องดำเนินการเองซึ่งมีหลักการว่า จะต้องพยายามแยกโปรแกรมออกเป็นส่วน ๆ คือเป็นโมดูล (Module) แต่ละโมดูลเป็นอิสระต่อกัน และมีทางเข้าทางเดียวทางออกทางเดียว (จะกระโดดเข้าไปในโมดูล หรือกระโดดออกมาจากกลางโมดูลไม่ได้) ภายในโมดูลอาจจะเป็นการปฏิบัติที่เรียงลำดับหรือตามเงื่อนไข หรือเป็นวงรอบหรือซ้อนกันอยู่ ดังที่จะกล่าวต่อไปนี้

โครงสร้าง IF

โครงสร้าง IF มีอยู่ 2 แบบคือ IF-THEN กับ IF-THEN-ELSE มีที่ใช้มากในทุกภาษา จะขอนำ IF-THEN-ELSE มาแสดง เพราะมีขั้นตอนมากกว่า ซึ่งมีรูปแบบเป็น

IF เงื่อนไขแรกเป็นจริง **THEN** (ให้)
  กระทำงาน 1
ELSE (หากไม่เป็นจริง ให้)
  กระทำงาน 2
  ENDIF

จะเห็นว่าโครงสร้าง IF มีทางเข้าตรง IF (ทางเดียว) และทางออกตรง ENDIF ทางเดียว เพื่อแสดงถึงการเขียนโปรแกรมโดยใช้โครงสร้าง IF จะสมมติว่าต้องการเขียนโปรแกรมให้ผู้ใช้ป้อนเลข 2 จำนวน นำมาหาผลต่างโดยนำเลขจำนวนน้อยไปลบเลขจำนวนมากเสมอ แล้วแสดงผลจะเขียนเป็นร่างโปรแกรมได้ดังนี้

```
BEGIN
  get(num1)
  get(num2)
  IF num1 > num2 THEN
    a1 := num1 - num2
  ELSE
    a1 := num2 - num1
  ENDIF
  Write(a1)
END
```

โปรดทราบว่าคำที่เราใช้ในร่างโปรแกรมเป็นคำสมมติเพื่อสื่อความหมายให้สั้นเท่านั้น เช่น get หมายถึงอ่านค่า ส่วน Write หมายถึงให้แสดงผลทางจอภาพ ที่เหลือเป็นตัวแปรสมมติคือ num 1, num 2 และ al นำไปเขียนเป็นโปรแกรมภาษาเครื่องได้ดังโปรแกรม DemoIf ในรูปที่ 1

แต่ในการเขียนโปรแกรมจริงผมได้เพิ่มเครื่องประดับคือ ให้แสดงคำเตือนว่า Enter two number 0 to 9 และเมื่อผู้ใช้กดคีย์แรกแล้วให้แสดงเครื่องหมายลบ เมื่อผู้ใช้กดคีย์ที่สองให้แสดงเครื่องหมายเท่ากับ และคำตอบ การที่กำหนดตัวเลขไว้ไม่ให้เกิน 9 เพื่อให้ตัวอย่างนี้ง่าย เพราะถ้าให้ป้อนเลขจำนวนมากกว่านี้จะต้องเขียนโปรแกรมเปลี่ยนเลขฐานสองเป็นสตริงตัวเลขจึงจะนำออกแสดงได้

ขอให้ดูในส่วนที่กำหนดข้อมูลคือ data SEGMENT ได้กำหนดให้ mssg, minus และ equal เป็นตัวแปรแบบสตริง และได้ให้ค่าเริ่มต้นไว้ดังที่เห็น ส่วน num1 และ num2 เป็นตัวแปรขนาด 1 ไบท์

ตัวโปรแกรมอยู่ใน code SEGMENT เริ่มต้นที่เลเบล prg ด้วยการให้ค่าดาต้าเซกเมนต์แก่ DS สำหรับโปร-

* รองหัวหน้าฝ่ายวิชาการ สถาบันวิชาการทหารเรือชั้นสูง (ส.ร.ส.)

แกรมของเราจริง ๆ เริ่มตรง main ตอนต้นจะเป็นการแสดงข้อความใน mssg อ่านค่า num 1 แสดงเครื่องหมายลบ อ่านค่า num 2 แล้วแสดงเครื่องหมายเท่ากับ

ต่อไปเป็นโครงสร้าง IF คือตรง ENTRY ทำการเปรียบเทียบค่าทั้งสอง แต่เราจะเปรียบเทียบค่าในหน่วยความจำทั้งคู่โดยตรงไม่ได้ ต้องโหลดค่าหนึ่งมาไว้ในรีจิสเตอร์ในโปรแกรมคือ AL เมื่อเปรียบเทียบกันจะเกิดการเปลี่ยนแปลงในแฟลก C และ Z เพื่อใช้คำสั่งกระโดดอย่างมีเงื่อนไข

โดยใช้คำสั่ง JB ELSE คือถ้า AL (ซึ่งก็คือ num 1) น้อยกว่า num 2 ให้กระโดดไปที่ ELSE ถ้าไม่น้อยกว่านั่นคือมากกว่าหรือเท่ากัน ให้ทำคำสั่งต่อไปคือ ลบ AL ด้วย num 2 แล้วกระโดดไป EXIT ส่วนเมื่อน้อยกว่านี้ให้กระโดดมาที่ ELSE ซึ่งจะลบ num 2 ด้วย num 1 ได้ผลลัพธ์ใน AL และออกจากโครงสร้าง IF ตรง EXIT เช่นกัน บวกด้วยค่า 30 H เพื่อให้เป็น ASCII แล้วนำออกจอภาพ จบโปรแกรมของเราตรง endmain ต่อไปนั้นเป็นการกลับไปยัง DOS ตามข้อกำหนดของ DOS

จะเห็นว่าโครงสร้าง IF โดยเนื้อแท้เมื่อเขียนด้วยภาษาเครื่อง จะตรวจสอบเงื่อนไขตรงทางเข้า (ENTRY) แยกกรณีที่เป็นเท็จไปทำตรง ELSE แล้วออกจากโครงสร้างตรงทางออก (EXIT) สำหรับกรณีที่เป็นจริงจะทำคำสั่งที่ตามมาแล้วกระโดดข้าม ELSE ไปที่ทางออก (EXIT) แรก ๆ อาจดูสับสนขอให้ยึดหลักที่กล่าวมานี้

```
; Program DemoIf  (DEMOIF.ASM)

INCLUDE  KBD&SCR.MCR
;PURGE    gotoXY, WhereXY, WriteCh, ReadKBD, ReadCh
;PURGE    WriteMssg, ReadMssg, TestKbd, nl

space    EQU   ' '

data     SEGMENT
mssg     DB    'Enter two number 0 to 9',13,10,'$'
minus    DB    ' - $'
equal    DB    ' = $'
num1     DB    ?
num2     DB    ?
data     ENDS

code     SEGMENT
         ASSUME  CS:code, DS:data

prg:     mov     ax, data    ; set DS to data segment
         mov     ds, ax

main:                        ;BEGIN
         WriteMssg  mssg
         ReadCh  num1        ; get(num1)
         WriteMssg  minus
         ReadCh  num2        ; get(num2)
         WriteMssg  equal
ENTRY:   mov     al, num1    ; IF num1 >= num2 THEN
         cmp     al, num2
         jb      ELSE
         sub     al, num2    ;   al := num1 - num2
         jnc     EXIT
ELSE:    mov     al, num2    ; ELSE
         sub     al, num1    ;   al := num2 - num1
EXIT:                        ; ENDIF
         add     al, 30h     ; convert to ASCII
         WriteCh  al         ; Write(al)
endmain:                     ;END

         mov     ah, 4Ch     ; return to DOS
endprg:  int     21h

code     ENDS
         END
```

รูปที่ 1 โปรแกรม DemoIf แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง IF

โครงสร้าง WHILE

โครงสร้าง WHILE เป็นโครงแบบทำงานในวงรอบ (loop) โดยตรวจสอบเงื่อนไขก่อน หากเงื่อนไขเป็นจริงจะทำงานในวงรอบจนกว่าเงื่อนไขเป็นเท็จจะออกจากวงรอบ มีรูปแบบเป็น

WHILE เงื่อนไขเป็นจริง **DO**
กระทำงาน
การเปลี่ยนแปลงเงื่อนไข
ENDWHILE

จะเห็นได้ว่าวงรอบ WHILE มีทางเข้าทางเดียวคือตรง WHILE และทางออกทางเดียวคือตรง ENDWHILE แต่เมื่อนำมาเขียนเป็นภาษาเครื่องจะมีลักษณะเป็น

ENTRY : ตรวจสอบเงื่อนไข
ถ้าเป็นเท็จไปยัง EXIT
ถ้าเป็นจริงให้ทำงาน
เปลี่ยนแปลงเงื่อนไข
กลับไปยัง ENTRY
EXIT :

เพื่อแสดงถึงการเขียนโปรแกรมโดยใช้โครงสร้าง WHILE ผมจะเขียนโปรแกรมให้ผู้ใช้ป้อนข้อความ 1 ประโยค ซึ่งจะต้องจบด้วยจุดแล้วกดคีย์ ENTER โปรแกรมจะอ่านประโยคนี้มาเก็บไว้ในตัวแปรสตริง แล้วนำออกแสดงซ้ำโดยนำออกแสดงทีละอักขระเรียงกันไป โดยนำเอาจุดมาเป็นเงื่อนไขว่าจบประโยค จะเขียนเป็นร่างโปรแกรมได้ว่า

```
BEGIN
  ReadString(str)
  PointIndexToFirstChar(idx)
  WHILE  str[idx] <> '.'  DO
    Write( str[idx] )
    idx := idx + 1
  ENDWHILE
END.
```

ดังได้กล่าวแล้วว่าคำสั่งและตัวแปรพวกนี้ผมเพียงแต่สมมติขึ้นเท่านั้น ReadString (str) หมายถึงให้อ่านข้อความที่เป็นสตริงของอักขระจนกว่าผู้ใช้จะกดคีย์ ENTER ให้แก่ตัวแปรสตริง str ส่วน PointIndexToFirstChar (idx) หมายถึงให้ค่าอินเด็กซ์ แก่ idx เพื่อชี้ไปยังอักขระตัวแรกในสตริงอะไรก็ได้ แต่ในที่นี้จะใช้กับตัวแปรสตริง str ดังนั้น หาก idx มีค่าเป็นศูนย์ str (idx) ก็คือ str (0) จะหมายถึงอักขระตัวแรก str (1) หมายถึงตัวที่สอง และต่อ ๆ ไป

นำไปเขียนเป็นโปรแกรมภาษาแอสเซมบลีได้ดังโปรแกรม DemoWhile ในรูปที่ 2 และได้เพิ่มคำเตือนอีกเช่นกัน โดยให้ไว้ในตัวแปรสตริง mssg จะเห็นว่าเราสามารถกำหนดตัวแปรสตริงให้ยาวกี่บรรทัดก็ได้ ส่วนตัวแปรสตริง str เป็นตัวแปรที่เตรียมไว้รับค่าจากคีย์บอร์ด แต่ได้ให้ค่าเริ่มต้นไว้เป็นเครื่องหมาย : ทั้ง 80 ไบท์ ส่วนไบท์สุดท้ายคือไบท์ที่ 81 ได้ให้ค่าไว้เป็นจุด เผื่อผู้ใช้ลืมกดคีย์จุด การทำงานในวงรอบ WHILE จะได้หยุดตรงนี้

ขอเว้นกล่าวถึงส่วนขึ้นต้นและลงท้ายของโปรแกรมซึ่งจะเหมือนกันทุกโปรแกรม สำหรับโปรแกรมนี้เริ่มต้นตรง main โดยใช้แมคโคร WriteMssg แสดงข้อความใน mssg ออกทางจอภาพ แล้วใช้แมคโคร Read Mssg อ่านข้อความจากคีย์บอร์ดมาให้แก่ตัวแปรสตริง str ซึ่งผู้ใช้จะป้อนได้ไม่เกิน 78 อักขระ เสร็จแล้วให้ค่าอินเด็กซ์ SI เป็น 2 เพราะจาก ReadMssg จะได้อักขระแรกที่ช่องนี้ (ไม่ใช่ที่ศูนย์)

เข้าวงรอบ WHILE ตรง ENTRY ดำเนินการตามที่กล่าวมาคือตรวจสอบเงื่อนไข ว่าอักขระตรง str (si) เป็น stop คือ จุดหรือไม่ ถ้าใช่ (เงื่อนไขเป็นเท็จ) ให้ออกจากวงรอบคือไปยัง EXIT ถ้าไม่ใช่ (เงื่อนไขเป็นจริง) ให้ทำงานในวงรอบคือ นำอักขระนี้ออกจอภาพ และเพิ่มค่าอินเด็กซ์ SI ขึ้นหนึ่ง เพื่อชี้ไปยังอักขระตัวถัดไป เสร็จแล้วให้กลับไปที่ ENTRY ใหม่

จะเห็นว่าโครงสร้าง WHILE มีทางเข้าทางเดียวคือตรง ENTRY และทางออกทางเดียวคือตรง ENTRY นั่นเอง แต่ให้กระโดดไปที่ EXIT คือท้ายของโมดูล จึงถือว่า EXIT เป็นทางออก

โครงสร้าง UNTIL

โครงสร้าง UNTIL เป็นโครงสร้างแบบทำงานในวงรอบเช่นกัน แต่จะทำงานก่อนแล้วตรวจสอบเงื่อนไข หากเงื่อนไขเป็นเท็จจะทำงานในวงรอบ จนกว่าเงื่อนไขเป็นจริงจึงจะออกจากวงรอบ มีรูปแบบเป็น

REPEAT
กระทำงาน
การเปลี่ยนแปลงเงื่อนไข
UNTIL เงื่อนไขเป็นจริง

แต่เมื่อนำมาเขียนในภาษาเครื่องจะมีลักษณะเป็น

ENTRY : เปลี่ยนแปลงเงื่อนไข
กระทำงาน
ตรวจสอบเงื่อนไข
หากเป็นเท็จ กลับไป **ENTRY**

```
; Program DemoWhile  (DEMWHILE.ASM)

INCLUDE  KBD&SCR.MCR
;PURGE    gotoXY, WhereXY, WriteCh, ReadKBD, ReadCh
;PURGE    WriteMssg, ReadMssg, TestKbd, nl

stop     EQU   '.'

data     SEGMENT
mssg     DB    'Enter a sentence end with period and ENTER',13,10,
         DB    'I will display it again.',13,10,'$'
str      DB    80 DUP (':')
         DB    '.'
data     ENDS

code     SEGMENT
         ASSUME  CS:code, DS:data

prg:     mov   ax, data   ; get data segment
         mov   ds, ax

main:                          ;BEGIN
         WriteMssg  mssg       ;  WritePrompt
         ReadMssg  77, str     ;  ReadString(str)
         nl                    ;  skip2line
         nl
         mov   si, 2           ;  PointIndexToFirstChar(idx)
ENTRY:   cmp   str[si], stop   ;  WHILE str[idx] <> '.' DO
         je    EXIT
         WriteCh  str[si]      ;    Write (str[idx])
         inc   si              ;    idx := idx + 1
         jmp   SHORT ENTRY
EXIT:                          ;  ENDWHILE
endmain:                       ;END

         mov   ah, 4Ch         ; return to DOS
endprg:  int   21h


code     ENDS
         END
```

รูปที่ 2 โปรแกรม DemoWhile แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง WHILE

EXIT : (เมื่อเป็นจริง)

ที่เขียนโครงสร้างในแบบนี้คือให้เปลี่ยนแปลงเงื่อนไขก่อนก็เพื่อให้ตรงกับความหมายของ UNTIL คือจะทำงานในวงรอบจนกว่าเงื่อนไขจะเป็นจริง ดังเช่นในโปรแกรม DemWhile จะไม่พิมพ์จุด แต่ถ้าใช้ UNTIL จะพิมพ์จุดออกไปด้วย

เพื่อเป็นการเปรียบเทียบการใช้โครงสร้าง UNTIL กับ WHILE จะนำโปรแกรมเดิมของ WHILE มาเขียนในรูปแบบของ UNTIL จะเขียนเป็นร่างโปรแกรมได้ดังนี้

```
BEGIN
  WritePrompt
  ReadString(str)
  PointIndexBeforeFirstChar(idx)
  REPEAT
    idx := idx + 1
    Write( str[idx] )
  UNTIL  str[idx] = '.'
END.
```

นำไปเขียนเป็นโปรแกรมภาษาแอสเซมบลีได้ดังโปรแกรม DemoUntil ในรูปที่ 3 ซึ่งได้เขียนเลียนแบบโปรแกรม DemoWhile จนถึง ENTRY จะเห็นว่าให้ทำงานในวงรอบก่อนคือ เพิ่มค่าอินเด็กซ์และนำอักขระในสตริง str ตรงอินเด็กซ์ไปออกจอภาพ เสร็จแล้วจึงตรวจสอบเงื่อนไขว่าอักขระตรง str (si) เป็น stop คือจุดหรือไม่ถ้าใช่(เงื่อนไขเป็นจริง) ให้ออกจากวงรอบ ตรง EXIT ถ้าไม่ใช่(เงื่อนไขเป็นเท็จ) ให้กลับไปที่ ENTRY ใหม่

รูปที่ 3 โปรแกรม DemoUntil แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง UNTIL

```
A>
; Program DemoUntil  (DEMUNTIL.ASM)

INCLUDE  KBD&SCR.MCR
;PURGE    gotoXY, WhereXY, WriteCh, ReadKBD, ReadCh
;PURGE    WriteMssg, ReadMssg, TestKbd, nl

stop     EQU   '.'

data     SEGMENT
mssg     DB    'Enter a sentence end with period and ENTER',13,10,
         DB    'I will display it again.',13,10,'$'
str      DB    80 DUP (':')
         DB    '.'
data     ENDS

code     SEGMENT
         ASSUME  CS:code, DS:data

prg:     mov    ax, data   ; get data segment
         mov    ds, ax

main:    WriteMssg  mssg       ; WritePrompt
         ReadMssg  78, str     ; ReadString(str)
         nl
         nl
         mov    si, 1          ; PointIndexBeforeFirstChar(idx)
ENTRY:                         ; REPEAT
         inc    si             ;   idx := idx + 1
         WriteCh  str[si]       ;   Write(str[idx])
         cmp    str[si], stop  ; UNTIL letter = stop
         jne    ENTRY
EXIT:
endmain:

         mov    ah, 4Ch    ; return to DOS
endprg:  int    21h


code     ENDS
         END
```

โครงสร้าง FOR

โครงสร้าง FOR เป็นการทำงานในวงรอบโดยกำหนดจำนวนรอบไว้แน่นอน มีรูปแบบเป็น

FOR ค่าต้น TO ค่าสุดท้าย step N DO
กระทำงาน
ENDFOR

โดยที่ค่าต้นและค่าสุดท้ายจะเป็นเลขจำนวนหรืออักขระก็ได้ ส่วน step N หมายถึงจำนวนที่นำมาเพิ่มแก่ค่าต้น ซึ่งอาจเป็นเลขลบจึงหมายถึงที่นำมาลดค่าต้นด้วย เมื่อนำมาเขียนเป็นโปรแกรมภาษาเครื่องจะมีลักษณะเป็น

ENTRY : ให้ค่าต้นแก่ตัวแปร
AGAIN : กระทำงาน
เพิ่ม (ลด) ค่าต้น
ตรวจว่าเกินค่าสุดท้ายหรือยัง
หากยังกลับไปที่ **AGAIN**
EXIT : (เมื่อค่าต้นเกินค่าสุดท้าย)

โปรแกรมตัวอย่างเพื่อแสดงการใช้ FOR โปรแกรมแรกจะให้แสดงอักษร A ถึง Z บนจอภาพ ซึ่งเขียนเป็นร่างโปรแกรมได้ดังนี้

```
BEGIN
  FOR al := 'A' TO 'Z' DO
    Write(al)
  ENDFOR
END.
```

นำไปเขียนเป็นโปรแกรมภาษาแอสเซมบลีได้ดังโปรแกรม DemoFor ในรูปที่ 4 ซึ่งรูปแบบของโปรแกรมคงไม่ต้องการคำอธิบายเพิ่ม

แต่ความเป็จริงแล้วในเครื่องไอบีเอ็มจะมีภาพของทุกรหัส ASCII จาก 00h ถึง 0FFh คือ 0 ถึง 255 หากเราต้องการภาพเหล่านี้จะต้องดำเนินการกับจอภาพโดยตรง โดยที่เราทราบว่าหน่วยความจำที่เป็นจอภาพอยู่ที่แอดเดรส B000:0000 ถึง B000:0F9F ในการนำภาพ ASCII ออกแสดงผมจะให้แสดงบรรทัดละ 16 ตัว 16 บรรทัด รวม 256 ตัว เขียนเป็นโปรแกรมภาษาแอสเซมบลีได้ดังโปรแกรม DemoFor 2 ในรูปที่ 5

ในโปรแกรมนี้ที่เพิ่มมาคือ ได้กำหนดให้ scrseg เป็น SEGMENT ที่ B000 และได้กำหนดตัวแปร scrn ไว้ที่ต้นเซกเมนต์นี้ โดยให้มีขนาด 2000 word คือ 4000 ไบท์ คือคลุมทั้งจอภาพ (160 x 25 = 4000) และในโปรแกรมได้กำหนดให้ ES แทน เซกเมนต์นี้ ดังนั้นในโปรแกรมเมื่ออ้างถึงตัวแปร scrn จะเป็นการอ้างต่อ ES (แต่ในโปรแกรมภาษาแอสเซมบลีจะดูไม่ออก)

ในการแสดงภาพออกจอภาพนี้ ผมจะใช้ BX เป็นฐาน คือถ้า BX มีค่าเป็น 0 จะหมายถึงบรรทัดแรก ถ้ามีค่าเป็น 160 หมายถึงบรรทัดที่ 2 และต่อ ๆ ไปจนถึง 2560 หมายถึงบรรทัดที่ 16 และในแต่ละบรรทัดจะใช้ DI เป็นอินเด็กซ์ โดยที่การแสดงภาพ 1 อักขระต้องการหน่วยความจำ 2 ไบท์ ดังนั้นเมื่อ DI มีค่าเป็น 0 จะหมายถึงตัวแรกในบรรทัด ถ้ามีค่าเป็น 2 หมายถึงตัวที่ 2 และต่อ ๆ ไปจนถึง 64 จะหมายถึงตัวที่ 32 ในบรรทัด ผมได้กล่าวไว้ว่าจะแสดงภาพอักขระ 16 ตัวต่อบรรทัด ที่กำหนดไว้ถึง 32 ตัว เพราะผมต้องการให้แสดงภาพ space คือวรรคคั่นระหว่างอักขระแต่ละตัว

การที่อักขระ 1 ตัว ต้องการหน่วยความจำที่เป็นจอภาพ 2 ไบท์ เพราะ

```
; Program  DemoFOR  (DEMFOR.ASM)

code    SEGMENT
        ASSUME  CS:code, DS:code

prg:

main:
ENTRY:  mov     al, 'A'     ; FOR al := 'A' TO 'Z' DO
AGAIN:  mov     dl, al      ;    Write(al)
        mov     ah, 02h
        int     21h
        inc     al
        cmp     al, 'Z'
        jbe     AGAIN
EXIT:                       ; ENDFOR
endmain:
        mov     ah, 4Ch     ; return to DOS
endprg: int     21h


code    ENDS
        END
```

รูปที่ 4 โปรแกรม DemoFor แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง FOR

```
; Program  DemoFor2   (DEMFOR2.ASM)

INCLUDE  KBD&SCR.MCR
;PURGE    gotoXY, WhereXY, WriteCh, ReadKBD, ReadCh
;PURGE    WriteMssg, ReadMssg, TestKbd, nl

data    SEGMENT
char    DW  0700h       ; Attribute and First ASCII code
space   DW  0720h       ; Attribute and Space
data    ENDS

scrseg  SEGMENT  AT 0B000h
scrn    DW  2000 DUP (?)
scrseg  ENDS

code    SEGMENT
        ASSUME  CS:code, DS:data, ES:scrseg

prg:    mov     ax, data        ; Set data segment
        mov     ds, ax
        mov     ax, 0B000h      ; Set extra segment
        mov     es, ax

main:   ClrScr                  ; Clear screen
        mov  bx, 0              ; FOR bx := 0 TO 2560 STEP 160 DO
FOR1:   mov  di, 0              ;   FOR di := 0 TO 64 STEP 2 DO
FOR2:   mov  ax, char           ;     Display(char)
        mov  scrn[bx + di], ax  ;
        inc  char               ;     char := Succ(char)
        inc  di                 ;     di := di + 2
        inc  di                 ;
        mov  ax, space          ;     Display(space)
        mov  scrn[bx + di], ax  ;
        inc  di                 ;     di := di + 2
        inc  di                 ;
        cmp  di, 64             ;
ENDFOR2: jb  FOR2               ;   ENDFOR
        add  bx, 160            ;   bx := bx + 160
        cmp  bx, 2560           ;
ENDFOR1: jbe  FOR1              ; ENDFOR
        GotoXY  20, 2           ; SetcurPosition
endmain:

        mov     ah, 4Ch     ; return to DOS
endprg: int     21h


code    ENDS
        END
```

รูปที่ 5 โปรแกรม DemoFor 2 แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง FOR ซ้อนกัน 2 รอบ

ไอบีเอ็มได้ออกแบบไว้เช่นนี้คือ ให้ไบท์แรกเป็นรหัส ASCII ไบท์ที่สองเป็น attribute (ทำให้เกิดเป็นภาพปกติ ภาพกลับ ภาพกะพริบ ตัวเน้น) ขอให้ดูการกำหนดตัวแปร Char ใน data เซกเมนต์ ค่า 07h คือค่า attribute ให้ภาพปกติ ส่วนค่า 00h เป็นรหัสแรกของ ASCII สำหรับตัวแปร space ค่า 20 h คือวรรค คือให้ภาพมืด 1 อักขระ

ขอให้สังเกตการกำหนดตัวแปรด้วยว่ากำหนดด้วย DW ดังนั้นเมื่อให้ค่า 0700h จะได้ว่า 00h เป็นไบท์ต่ำ และ 07h เป็นไบท์สูง หากกำหนดด้วย DB จะต้องให้ค่าเป็น 00h, 07h ที่กำหนดเป็น DW เพื่อเตรียมไว้สำหรับการดำเนินกรรมวิธี 16 บิท โดยตรง

โปรแกรมเริ่มต้นด้วยการลบจอภาพเสร็จแล้วทำวงรอบ FOR สองวงซ้อนกัน วงนอกเป็นบรรทัด วงในเป็นตำแหน่งในบรรทัด ในวงรอบก็เพียงแต่นำภาพอักขระไปออกจอภาพโดยตรง ทำโดยให้ค่ารหัสอักขระนั้นแก่ AX ก่อน แล้วนำไปไว้ที่หน่วยความจำจอภาพโดยตรง โดยใช้คำสั่งว่า mov scrn (bx + di), ax เสร็จแล้วเพิ่มค่ารหัสอักขระขึ้นหนึ่งเป็นตัวถัดไป เพิ่มสองแก่อินเด็กซ์ นำภาพ space ออกจอภาพ เพิ่มสองอินเด็กซ์ใหม่ แล้วตรวจว่าจบบรรทัดหรือยัง ถ้ายังกลับไปทำวงในซ้ำ

หากจบบรรทัดจะเพิ่มค่าฐานอีก 160 แล้วตรวจดูว่าเกิน 16 บรรทัดหรือยัง ถ้ายังกลับไปทำวงนอกซ้ำ ซึ่งจะให้อินเด็กซ์เป็นศูนย์ใหม่ ซึ่งจะทำวงในซ้ำอยู่ 16 เที่ยว แล้วทำวงนอกซ้ำอยู่ 16 เที่ยวเช่นกัน ถึงหลุดจากวงรอบ FOR ทั้งสอง ได้ภาพของรหัส ASCII ทั้ง 256 ตัวบนจอภาพ เสร็จแล้วผมได้ให้เคอร์เซอร์ไปอยู่บรรทัดที่ 20 คอลัมน์ที่ 2 เพื่อไม่ให้เกะกะ

ในการทดลองรันโปรแกรมนี้ คุณที่มีคาแรกเตอร์เยนฯภาษาไทยขอให้สวิตช์ไปที่อักษรไทยจะเห็นภาพอักษรไทย คุณสามารถสวิตช์กลับไปกลับมาได้ ซึ่งอาจจะช่วยให้เข้าใจในเรื่องของคาแรกเตอร์เยนฯเพิ่มขึ้น

โครงสร้าง CASE

โครงสร้าง CASE เป็นโครงสร้างแบบเลือกปฏิบัติเช่นเดียวกับ IF แต่มีกรณีให้เลือกหลายกรณีมีรูปแบบเป็น

```
CASE    กรณี    OF
1 :     กระทำงาน    1
2 :     กระทำงาน    2
.         .         .
n :     กระทำงาน    n
ENDCASE
```

ตัวเลข 1 ถึง n ไม่ได้หมายความว่าจะต้องเป็นตัวเลขเสมอไป อาจเป็นอักขระหรืออื่น ๆ ก็ได้ และการเขียนโครงสร้าง CASE ในภาษาแอสเซมบลีก็เขียนได้หลายแบบ ที่จะนำมาแสดงนี้เป็นแบบหนึ่งเท่านั้น โดยสมมติว่าแสดงเมนูให้ผู้ใช้เลือกโดยกดคีย์เลข 0 ถึง 5 เมื่อผู้ใช้กดคีย์ใด ก็ให้ทำงานในโปรแกรมย่อยของคีย์นั้น ๆ แต่ในที่นี้จะเพียงแต่ให้พิมพ์ข้อความออกมาเท่านั้น

เขียนเป็นโปรแกรมภาษาแอสเซมบลีได้ดังโปรแกรม DemoCase ในรูปที่ 6 ได้พิมพ์ร่างโปรแกรมกำกับไว้ด้วยแล้ว ตอนต้นใช้วงรอบ REPEAT อ่านคีย์มาตรวจว่าเป็นคีย์ 0 ถึง 5 หรือเปล่า ถ้าผู้ใช้กดคีย์อื่นจะวนอยู่ในวงรอบนี้ ต่อเมื่อผู้ใช้กดคีย์ 0 ถึง 5 คีย์ใดคีย์หนึ่งจึงจะออกจากวงรอบ พิมพ์ภาพของคีย์ที่กดทำให้เกิดความรู้สึกว่าเครื่องไม่รับคีย์อื่น

```
; Program  DemoCase  (DEMCASE.ASM)

INCLUDE  KBD&SCR.MCR
;PURGE    gotoXY, WhereXY, WriteCh, ReadKBD, ReadCh
;PURGE    WriteMssg, ReadMssg, TestKbd, nl

data     SEGMENT
mssg     DB    'Your option is 0 to 5: $'
mssg0    DB    ' THIS IS ZERO $'
mssg1    DB    ' THIS IS ONE $'
mssg2    DB    ' THIS IS TWO $'
mssg3    DB    ' THIS IS THREE $'
mssg4    DB    ' THIS IS FOUR $'
mssg5    DB    ' THIS IS FIVE $'

TABLE    DW  SBR0
         DW  SBR1
         DW  SBR2
         DW  SBR3
         DW  SBR4
         DW  SBR5
data     ENDS

code     SEGMENT
         ASSUME  CS:code, DS:data

prg:     mov    ax, data       ; Set data segment
         mov    ds, ax

main:                          ;BEGIN
         WriteMssg  mssg       ;  Write(mssg)
rep:     mov    ah, 08h        ;  REPEAT
         int    21h            ;    Read(KBD, char)
         cmp    al, 30h        ;
```

รูปที่ 6 โปรแกรม DemoCase แสดงการเขียนโปรแกรมโดยใช้โครงสร้าง CASE

```
        jb    rep
        cmp   al, 35h
until:  ja    rep           ; UNTIL char IN [0..5]
        WriteCh  al         ; Write(al)
        sub   al, 30h       ; char := int(char)
case:   xor   ah, ah        ; CASE al OF
        sal   ax, 1
        mov   di, ax
        jmp   TABLE[di]
SBR0:   WriteMssg mssg0     ; '0' :  DoZero
        jmp   SHORT endcase
SBR1:   WriteMssg mssg1     ; '1' :  DoOne
        jmp   SHORT endcase
SBR2:   WriteMssg mssg2     ; '2' :  DoTwo
        jmp   SHORT endcase
SBR3:   WriteMssg mssg3     ; '3' :  DoThree
        jmp   SHORT endcase
SBR4:   WriteMssg mssg4     ; '4' :  DoFour
        jmp   SHORT endcase
SBR5:   WriteMssg mssg5     ; '5' :  DoFive
endcase:                    ; ENDCASE
endmain:                    ;END

        mov   ah, 4Ch   ; return to DOS
endprg: int   21h


code    ENDS
        END  prg
```

โปรแกรมที่แสดงโครงสร้าง CASE เริ่มตรง case เปลี่ยนค่าคีย์ที่อ่านมาซึ่งเป็น ASCII ไปเป็นเลขฐานสอง แล้วคูณด้วย 2 ให้แก่ DI เพื่อใช้ในการเปิดตารางกระโดด ขอให้สังเกตด้วยว่าได้ทำตารางกระโดดไว้ในดาต้าเซกเมนต์ ส่วนการกระโดดจริง ๆ ดำเนินการในโค้ดเซกเมนต์ แต่ละงานเมื่อทำงานเสร็จแล้วให้กระโดดไปที่ EXIT อันเป็นทางออกของโครงสร้าง CASE

โปรแกรมนี้ความจริงควรเขียนในแบบเรียกโปรแกรมย่อยมากกว่า แต่เพื่อให้เห็นโครงสร้างของ CASE ถนัดจึงเขียนในแบบกระโดด

ตรรก AND และ OR

ในการเขียนโปรแกรมเป็นโครงสร้างดังกล่าวมานี้ ในบางครั้งเราจำเป็นต้องใช้ตรรก AND หรือ OR เข้าร่วมด้วย เช่นกล่าวว่า

```
IF  (A = 5)  AND  (B = 9)  THEN
  num := num - 1
ENDIF
```

ความหมายคือ ให้ลดค่า num ลงหนึ่งถ้าทั้ง 2 กรณีนี้เป็นจริง คือต้องทั้ง A เท่ากับ 5 และ B เท่ากับ 9 ด้วย หากกรณีใดกรณีหนึ่งไม่เป็นจริง เช่น A ไม่เท่ากับ 5 ก็จะไม่มีการลดค่า num ตรรก-AND เช่นนี้ เขียนในภาษาปาสคาล และซี ได้โดยตรง แต่เมื่อนำมาเขียนในภาษาเครื่องจะต้องเขียนเป็น

```
IF  A = 5  THEN
  IF  B = 9  THEN
    num := num - 1
  ENDIF
ENDIF
```

ซึ่งเขียนเป็นโปรแกรมภาษาแอส-เซมบลี โดยสมมติว่าให้ AX เก็บค่าของ A และ BX เก็บค่าของ B ได้ดังนี้

```
IF1:  cmp  ax, 5
      jne  EIF1
IF2:  cmp  bx, 9
      jne  EIF2
      sub  num, 1
EIF2:
EIF1:
```

ในกรณีของตรรก OR คือถ้าต้องการให้ลดค่า num ลงหนึ่ง เมื่อกรณีใดกรณีหนึ่งเป็นจริง จะเขียนเป็นโปรแกรมในแบบของภาษาปาสคาลและซีได้เป็น

```
IF  (A = 5)  OR  (B = 9)  THEN
  num := num - 1
ENDIF
```

เมื่อนำไปเขียนในภาษาเครื่องจะต้องแปลงเป็น

```
IF  (A = 5)  THEN
  num := num - 1
ELSE
  IF  (B = 5)  THEN
    num := nuM - 1
  ENDIF
ENDIF
```

แต่เมื่อนำไปเขียนเป็นโปรแกรมจริงจะแปลงเป็น

```
IF1:  cmp  ax, 5
      je   ELSE
IF2;  cmp  bx, 9
      jne  ENDIF
ELSE: sub  num, 1
ENDIF:
```

สำหรับในโครงสร้าง WHILE และ UNTIL เช่นต้องการกำหนดเงื่อนไขว่า

WHILE (A = 5) AND (B = 9) DO

หรือ **UNTIL (A = 5) AND (B = 9)**

ให้นำเงื่อนไขของ IF มาใช้ เพราะความจริงแล้วในโครงสร้าง WHILE ก็คือ WHILE IF และโครงสร้าง UNTIL ก็คือ UNTIL IF